YÖNETİM BİLİŞİM SİSTEMLERİ DERGİSİ
http://dergipark.gov.tr/ybs

Yayın Geliş Tarihi: 14.02.2021
Yayına Kabul Tarihi: 02.06.2021
Online Yayın Tarihi: 07.07.2021

Cilt: 7, Sayı: 1, Yıl: 2021, Sayfa:10-23
ISSN: 2148-3752

BİLİŞİM SİSTEMİ YAZILIM GELİŞTİRME SÜRECİNDE KULLANILAN ŞELALE VE SCRUM YÖNTEMLERİNİN KARŞILAŞTIRILMASI VE KARMA YÖNTEMİN İNCELENMESİ

Dr. Doğan YILDIZ[1]

[1] Bilgi Teknolojileri Başkanlığı, Türk Havacılık ve Uzay Sanayii A.Ş. (TUSAŞ), Ankara Türkiye

ÖZET

Günümüzde özel işyerlerinde veya kamuda birçok işlem bilgi sistemi ile yapılmaktadır. Dolayısı ile bilgi sistemine olan talep çok artmıştır ve ilerleyen zamanda daha da çok artacaktır. Yönetim bilgi sistemi olsun başka sistem olsun geliştirme aşamalarında farklı yazılım yaşam döngüsünden geçerek son ürün haline gelmektedir. Yazılım yaşam döngüsünün 1970'li yıllarda ortaya konulan şelale yönteminden günümüzde yoğun bir şekilde kullanılan scrum yöntemine kadar çok farklı çeşidi bulunmaktadır. Ancak günümüzde bilgi sistemine olan talep arttığından dolayı bu talebe hızlı bir şekilde cevap verecek olan çevik yazılım geliştirme yöntemlerine doğru bir geçiş yaşanmaktadır. Bazı durumlarda da her yöntemin iyi yanını alıp karma yöntemler uygulamak daha faydalı olabilmektedir. Bu çalışmanın amacı farklı bilişim sistemi yazılım geliştirme yöntemlerinin karşılaştırılması ve bu yöntemlerin karması hakkında değerlendirmeler yapmaktır. Uygulama kısmında ise şelale yönteminden scrum yöntemine geçişte dikkat edilmesi gereken hususlar ele alınmıştır. Ayrıca, her iki yöntemin karması olan yöntem de değerlendirilmiştir.

Anahtar Kelimeler: Yönetim Bilişim Sistemleri, Çevik Yöntem, Scrum, Şelale Yöntemi, Karma Yöntem

COMPARISON OF WATERFALL AND SCRUM METHODS IN INFORMATION SYSTEM SOFTWARE DEVELOPMENT PROCESS AND EXAMINATION OF MIXED METHOD MANAGEMENT

ABSTRACT

Nowadays, many transactions are carried out with information systems in private workplaces or public places. Consequently, the demand for the information system has increased and will increase even more in the future. Be it the management information system or other system, it goes through a different software life cycle in the development stages and becomes the final product. There are many different types of software life cycle, from the waterfall method introduced in the 1970s to the scrum method, which is used extensively today. However, today, due to the increasing demand for the information system, there is a transition towards agile software development methods that will quickly respond to this demand. In some cases, it may be more beneficial to take the good side of each method and apply mixed methods. The aim of this study is to compare different information system software development methods and evaluate the mix of these methods. In the application part, the issues to be considered in the transition from the waterfall method to the scrum method are discussed. In addition, the method, which is a mixture of both methods, was also evaluated.

Keywords: Management Information Systems, Agile Method, Scrum, Waterfall Method, Mixed Method

GİRİŞ

Değişim, insan hayatının her zamanında ve her yerinde yer almaktadır. Değişim var ise gelişim vardır düşüncesi ile hayatımızdaki her şeyi değiştirmekteyiz. Standart halde tek düze bakış açısı ile iş yapanlar, belirli bir zaman sonrasında kendisini değiştirmek ve geliştirmek zorundadır. Süreçte veya üründe iyileştirme ve geliştirme çalışmaları yapılmaz ise zamanla süreç içinde darboğazlar oluşmaya başlar, ürün değişime ayak uyduramaz ise yok olmaya başlar ve sonuçta sürecin ve ürünün sahibi olan işletmenin kazancı azalır. Bilişim sistemi yazılımı geliştirme ömür devri süreçleride aynı şekilde bir değişim ve gelişim içindedir. Eskiden sadece kod geliştirme şeklinde yer alan yazılım geliştirme yaşam döngüsü zamanla farklı safhalarında eklenmesi ile daha süreçsel bir hale gelmiştir. Uzun bir süre şelale yöntemi ve farklı türevleri yazılım geliştirme yaşam döngüsünde baskın bir şekilde kullanılmıştır. Belirli bir zaman sonrasında bu yöntemde bazı yetersizlikler görülmüş ve bununla ilgili başka arayışlar içine girilmiştir. Bu arayışın en büyük nedenlerini, bilişimin daha çevik bir şekilde hareket etme zorunluluğu ve proje maliyetlerini düşürmek şeklinde açıklayabiliriz. Bunlar ve farklı nedenlerden dolayı çevik yazılım geliştirme yöntemleri ortaya çıkmıştır. Zamanla ilk çıkan çevik yöntemlere ilave olarak farklı alt yöntemler de ortaya çıkmıştır. Bunların içinden en çok kullanılan yöntem scrum'dır.

Yazılım geliştirme yaşam döngüsünde kullanılan çok farklı modeller vardır. Bunlar, kodla ve düzelt, şelale, V model, spiral model, artımlı geliştirme, farklı çevik yöntemler; XP (extreme programming), kanban, yalın yazılım geliştirme ve scrum'dır. Bu çalışma kapsamında özellikle zamanında çok yoğun olarak kullanılan şelale ve günümüzde çok yoğun olarak kullanılan scrum yöntemleri hakkında bilgiler sunulmuş ve her iki yönteminde olumlu ve olumsuz yönleri açıklanmıştır. Şelale yönteminden scrum yazılım geliştirme yaşam döngüsüne geçişte ele alınması gereken hususlar aktarılmıştır. Bu çalışma ile araştırmacılar her iki yöntem ile ilgili bilgi sahibi olabilecek, bunların arasındaki farkları ve projelerinde hangi yöntem ile ilerlerler ise ne ile karşılaşacaklarını görebileceklerdir. Ayrıca karma yöntem hakkında da bilgi sahibi olacak ve bu modeli projelerinde kullanabileceklerdir.

LİTERATÜR TARAMASI

Royce (1970), analiz ve kodlama safhalarından oluşan yazılım geliştirme ömür devrini, sistem gereksinimleri, yazılım gereksinimleri, analiz, program tasarımı, kodlama, test, işletim şeklinde sıralamıştır. Ayrıca, geliştirme riskini önlemek amacı ile de beş adımda özellikleri açıklanmıştır. Birinci adım: Program tasarımı ilk öncelikle gelmeli, bu adımda veri tabanı ve süreç tasarımı yapılması ve ayrıca yazılım tasarımı ile ilgili tasarımların da yapılması gerektiği açıklanmıştır. İkinci adım: Doküman tasarımı kapsamında ise bu safhalarda hangi dokümanların üretilmesi gerektiği açıklanmaktadır. Üçüncü adım: İki kere yapılması, bu kapsamda ilk çıkacak ürün son ürünün bir simülasyonu olacağını açıklamaktadır. Dördüncü adım: Testi planla, kontrol et ve izle şeklinde yer almaktadır ve testin önemi üzerinde durulmuştur. Beşinci adım: Müşteriyi dahil et kısmında ise müşterinin işin içine dahil edilmesinin önemi açıklanmıştır.

Boehm ve Turner (2003), çevik yazılım geliştirme manifestosu 2001 yılında oluşturulmuştur ve çevik yazılımın 12 prensibi vardır. Manifesto kapsamında; süreçler ve araçlardan ziyade bireyler ve etkileşimlere, kapsamlı dokümantasyondan ziyade çalışan yazılıma, sözleşme pazarlıklarından ziyade müşteri ile iş birliğine, bir plana bağlı kalmaktan ziyade değişime karşılık vermeye değer verdiklerini ifade etmişlerdir (Beck, Beedle ve Bennekum, 2001; Beck ve diğerleri, 2001). Çeviklik ve disiplin arasındaki beş kritik karar noktası vardır. Boyut, çevik yöntemlerin küçük ürünler ve takımlar için uygun olduğunu; kritiklik, kritik sistemler için disiplinli süreçlerin uygun olduğunu; dinamizm, çevik yöntemlerin çok dinamik olduğunu; personel, personelin çevik yöntemler için çevik ve belirli seviyede uzman olması gerektiği; kültür, çevik yöntemlerdeki personel özgüvenli ve kendini konforlu hissetmesi gerekir.

Palmquist, Lapham, Miller, Chick ve Ozkaya (2013) şelale ve çevik yöntemleri karşılaştıran çalışmalarında, çevik ve şelale yöntemlerinin özellikleri anlatılmış ve bunun yanında V model hakkında da bilgiler sunulmuştur. Çevik yazılım geliştirmede genel olarak ilgili proje kapsamında bir iş listesi olduğu ve bunun farklı yazılım sürümlerine (çıkımlar) bölündüğü ve her bir sürümün kendi iş listesi ve zaman planı olduğunu, her bir sürüm içinde de iterasyonlar olduğu ve her bir iterasyonunda

bir sprint ile çalışıldığı aktarılmıştır. Şelale ve çevik modellerin temelleri olarak birçok kriter sıralanmış ve ayrıca her iki modelin benzerlik ve farklılık açısından karşılaştırması yapılmıştır. Gencer ve Kayacan (2017), şelale ve çevik yöntem hakkında bilgiler sunmuşlardır. Her iki yöntemin tarihçesi, ilkeleri ve özellikleri ile kısıtları ve faydaları hakkında açıklamalarda bulunmuşlardır. Ayrıca, şelale ve çevik proje yönetimi ve yönetimin rolleri, proje görünürlüğü, proje karmaşıklığı, proje büyüklüğü, altyüklenici kullanımı, müşteri katılımı, girişimci organizasyonu, makine organizasyonu, profesyonel organizasyon, farklılaşmış organizasyon, yenilikçi organizasyon, proje elemanlarının rolleri, proje elemanlarının tecrübe ve yetenekleri, takım büyüklüğü, proje elemanlarının alan bilgisi, iş birliği, proje elemanlarının yerleşimi, ekip çalışması, yazılım nitelikleri, kalite güvence ve hata toleransı, değer üretimi, kullanıcı deneyimi, dokümantasyon ve modelleme, teslimatlar, yeniden kullanılabilirlik, kullanım onayı, ihtiyaca yabancılaşma, değişime açıklık, gereksinimlerin anlaşılırlığı, uygulanabilirliği ve güvenilirliği, gereksinim yönetimi, zaman, iş gücü, ve risk yönetimi maddeleri altında karşılaştırmışlar ve yazılım proje yönetimi metodolojisi hakkında önerilerde bulunmuşlardır. Vohra ve Singh (2013), klasik yöntemler ile çevik yöntemleri karşılaştırmışlardır. Bunun yanında ayrıca yazılım süreç iyileştirme yöntemleri olan ISO15504 ve CMMI modelleri ile çevik yöntemleri de karşılaştırmışlardır. Balaji ve Murugaiya (2012), şelale, V model ve çevik yazılım geliştirme yaşam döngüsü modellerinin karşılaştırmalarında her bir modeli açıklayıp, modellerin artıları ve eksilerini madde madde aktarmışlardır. Ayrıca, hangi proje türü için hangi yöntemin seçilmesi şeklinde önerilerde bulunmuşlardır. Buna göre, eğer gereksinim sık sık değişiyor, proje küçük, ürünü kısa sürede teslim etmek gerekiyor ve yetenekli kaynak da var ise çevik model seçilebilir. Eğer gereksinimler net ve proje büyük ise şelale yöntemi seçilebilir. Eğer gereksinim değişir, proje büyük, her aşamada uygun doğrulama varsa ve test çalışmaları geliştirmenin erken aşamalarında yer alıyorsa V modelin seçilebileceğini ifade etmişlerdir. Mahalakshmi ve Sundararajan (2013), her iki yöntemi tanıtarak yine her ikisininde olumlu ve olumsuz yönlerini aktarmışlardır. Davis ve Radford (2014), çevik yöntemler hakkında yanlış bilinenleri sıralamışlardır. Bunlar; gereksinimlerin alınması gereksizdir, çevik takımlar diğer modellere göre daha çok fonksiyon üretirler, çevik takımlar kendi kendine organizedir ve yönetilmeye ihtiyaçları yoktur, çevik yaklaşım tüm proje çeşitlerine uygulanabilir. Bunun yanında şelale yöntemi ile ilgili olarak da yanlış bilinenleri şu şekilde sıralamışlardır. Bunlar; yüzde yüz şelale yöntemi uygulayan proje, tüm projelere şelale yöntemi uyar, şelale yöntemi riski düşürür ve ürün dağıtım belirsizliklerini yok eder. Sommerville (2016), şelale yöntemini beş safhaya ayırmıştır. Bunlar, gereksinimlerin tanımlanması, sistem ve yazılım tasarımı, kodlama ve birim test, entegrasyon ve sistem testleri ve son olarak işletim ve bakımdır. Ayrıca şelale yönteminin bazı sistemler için zorunlu olarak kullanılması gerektiğini aktarmıştır. Bunlar; gömülü sistemler: Burada yazılım, donanım ile etkileşim halindedir. Kritik sistemler: Bu tür sistemlerde emniyet ve güvenlik ön plandadır ve bunlar ile ilgili detaylı bir şekilde yazılım spesifikasyonu hazırlamak gerekmektedir. Büyük yazılım sistemleri: Bu tür sistemlerde farklı işletmeler ortaklaşa çalışması gerekmektedir. Bunların yanında resmi olmayan takım iletişimi ve yazılım gereksinimleri çok çabuk değişiyor ise bu durumda iterasyonlu model veya çevik yöntemlerin kullanılmasının daha uygun olacağını ifade etmiştir.

Forrester (2011), bugün çoğu organizasyonda çevik yöntemlerin Water – Scrum – Fall şeklinde uygulanmakta olduğunu ifade etmiştir. Bunun anlamı, gereksinim belirleme ve sistem tasarımı aşaması şelale yöntemi ile yapılmakta, ortada kodlama ve test kısmı scrum metodolojisi ile yapılmakta, son kısım olan uygulamaya alma ve diğer kısımlarda şelale yöntemi ile ilerlenmektedir. Singhto ve Phakdee (2016), scrum ve şelale metodolojilerinin kombinasyonu ile Tayland'da küçük ve orta ölçekli hizmet ve üretim işletmeleri için yazılımın servis olarak sunulduğu bir uygulama çalışması sonucunda, kullanıcı ve müşteri memnuniyetinde artış, bilgi teknolojilerinde finansal kazanç, az zaman kaybı ve iş süreçlerinde denge şeklinde büyük fayda sağladıklarını göstermişlerdir. Bhavsar, Shah ve Gopalan (2020), çevik süreçlerin yenilenmesi kapsamında üç farklı yöntem, başta scrum, ortada kanban ve sonda şelale yöntemi olmak üzere entegre edilmiş ve sonuçta yöntemlerin tek başlarına kullanılmalarına göre büyük avantaj sağladıkları sonucuna varmışlardır.

Başar, Özkaya, Kesgin (2015) çalışmalarında bir teknoloji şirketindeki farklı yazılım geliştirme yöntemleri hakkında bilgiler sunmuşlardır. Öncelikle şelale yöntemi ve bu yöntemde karşılaşılan sorunlar ardından da scrum yöntemi ve karşılaşılan sorunlar hakkında bilgiler sunmuşlardır. Her iki yöntemde karşılaşılan sorunların üstesinden gelmek için kanban yönteminin

devreye alındığını ve onunla ilgili nelerin iyileştirilmeye başlandığı hakkında bilgi verilmiştir. Aydıner, Esen, Özlü (2020) çalışmalarında Türkiye'de farklı sektörlerdeki işletmelerin yazılım projelerinde çalışan ve farklı pozisyonlarda bulunan 105 kişi ile saha araştırması yapmışlardır. Sonuçta, scrum yönteminde insan başarı faktörleri, işletme süreç ve uygulamalarının başarı faktörleri ile organizasyona ilişkin başarı faktörlerini irdelemişlerdir. Ceylan, Gürsev (2020) yaptıkları çalışmada şelale, scrum ve kanban yöntemlerini karşılaştırmış ve sonucunda ilgili firmada klasik yönetim metotlarından şelale yönteminden verim alınamadığı ve firmanın başarılı bir proje yönetimi için scrum ve kanban gibi çevik yöntemleri kullanmasının daha etkili olacağı sonucuna varmışlardır. Macit ve Tüzün (2015) yaptıkları çalışmada şelale ve scrum modellerini karşılaştırmış ve sonuçta gereksinimleri ve teknolojisi bilinen bir iş yapılıyorsa şelale yöntemi tam tersi durumda ise scrum yönteminin kullanılmasını önermişlerdir. Kuhrmann, Diebold, MüNch, Tell, Garousi, Felderer, Trektere, McCaffery, Linssen, Hanser, Prause (2017) yaptıkları çalışmada şelale, scrum ve ötesinde hibrid yazılım ve sistem geliştirmede farklı yaklaşımların nasıl kombine edildiği ve buna hangi nedenlerin etkili olduğuna bakmışlardır. Farklı 69 kişi ile yaptıkları çalışmada farklı geliştirme yaklaşımlarının kullanıldığını görmüşlerdir. En önemlisi ise klasik yaklaşımın üzerine çevik yaklaşımların adopte edildiğini görmüşlerdir. Hibrid yaklaşımın şirket büyüklüğü ve dış etkenlerden bağımsız olduğunu bunun genellikle deneyim, bilgi ve uygulamadan kaynaklandığını ifade etmişlerdir. Stoica, Ghilic-Micu, Mircea, Uscatu (2016), çalışmalarının sonucunda her projenin kendine özgü dinamikleri olduğu ve buna bağlı olarak her projenin farklı geliştirme yöntemi kullanabileceğini ifade etmişlerdir. Andrei, Casu-pop, Gheorghe, Boiangiu (2019), çalışmalarında şelale yönteminin, iyi tanımlanmış küçük projeler ve değişmeyen gereksinimler olması durumunda daha iyi bir çözüm olacağını, çevik yönteminde ise sürekli teslimat ve geri bildirimin önemli olduğunu, gereksinimlerin iyi tanımlanmadığını ifade etmişlerdir.

Bu kapsamda yapılan anket ve raporlara baktığımızda ilk olarak Hewlett Packard (HP) Enterprise firmasının 2017 yılında çeviklik yeni normallik adlı çalışmasında 601 kişi ile yaptıkları kısa bir ankette yüzde 16 oranında sadece çevik yazılım geliştirdiklerini, yüzde iki oranında sadece şelale yöntemi kullandıklarını, yüzde yedi oranında şelale yöntemine yakın olduklarını, yüzde 51 çevik yöntemlere yatkın olduklarını ve yüzde 46 oranında ise karma metotlar kullandıklarını ifade etmişlerdir. Ayrıca, Agile Turkey tarafından yayımlanan sekizinci yıllık çeviklik raporu, Türkiye'de, katılımcılar yüzde 94'ü firmalarının çevik yaklaşımları kullandıklarını ifade etmişlerdir. Bunun yanında organizasyonlarında yüzde 90 scrum, yüzde 29 şelale ve yüzde 46 kanban metodunu kullandıklarını aktarmışlardır (katılımcıların çoklu seçme hakkı vardır). The Standish Group'un 2015 yılında 10.000'in üzerinde küçük, orta ve büyük çaplı yazılım projeleri ile oluşturdukları Chaos Raporuna göre çevik yazılım geliştirme yöntemleri ile ilerleyen yazılımlardaki başarısızlık oranı yüzde dokuz seviyelerinde iken bu oran şelale yöntemlerinde yüzde 29 seviyelerindedir. Aynı raporda, kısmen başarılı oranı çevik yöntemlerde yüzde 52, şelale yönteminde ise yüzde 60 seviyesindedir. Son olarak bu raporda projelerin başarılı olma durumları ise çevik yöntemlerde yüzde 39, şelale yöntemlerinde ise yüzde 11 seviyelerindedir. Aynı grubun 2018 yılında yayınladıkları raporda bu oranlar çevik yazılım projelerinde başarı yüzde 42, başarısızlık oranı ise yüzde sekizdir. Geriye kalan yüzde 50 oran ise ortadadır. Bu oranlara şelale yöntemi tarafından bakacak olursak yüzde 26 başarılı, yüzde 21 başarısız ve yüzde 53 ortadadır. Bu raporda bu başarısızlıkların nedenlerini ise, belirli ve tanımlı olmayan gereksinimler, değişen gereksinimler, gerçekçi olmayan beklentiler, yönetim desteğinin ve proje kaynağı yetersizlikleri şeklinde sıralanmıştır.

METODOLOJİ

Yazılım Geliştirme Süreci

Proje Yönetimi Bilgi Birikimi (Project Managemnet Body of Knowledge –PMBOK) Yazılım Uzantısı (Software Extension To The PMBOK) üç farklı şekilde yazılım geliştirme metodu anlatmaktadır. Bunlar; tahmin edici, tekrarlayıcı ve duruma göre uyabilen şeklindedir. Tahmin edici metot şelale yöntemini tariflemektedir. Tekrarlayıcı ise döngüsel veya tekrarlayıcı yazılım geliştirme yöntemlerini tariflemektedir. Duruma göre uyan ise çevik ve scrum yöntemlerini aktarmaktadır.

ISO/IEC/ IEEE 12207 Sistem ve Yazılım Mühendisliği – Yazılım Yaşam Döngüsü Süreçleri (Systems and Software engineering — Software Life Cycle Processes) standardı yazılım geliştirme

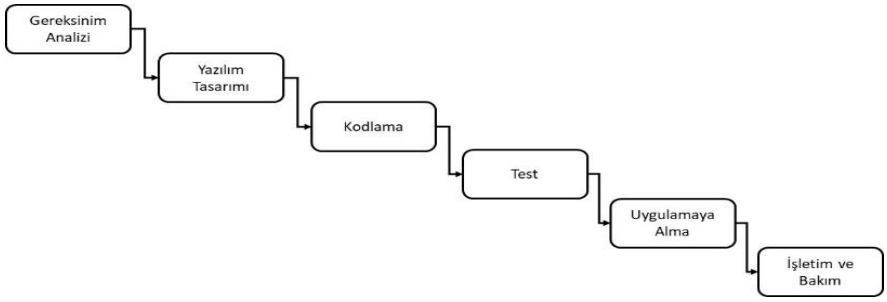
teknik süreçlerini, iş veya görev analizi süreci, paydaş ihtiyaç ve gereksinimleri tanımlama süreci, sistem/yazılım gereksinimleri tanımlama süreci, mimari tanımlama süreci, tasarım tanımlama süreci, sistem analiz süreci, uygulama süreci, entegrasyon süreci, doğrulama (test) süreci, uygulamaya geçiş süreci, yazılım kabul doğrulama (test) süreci, işletme süreci, bakım süreci, kullanımdan kaldırma süreci şeklinde tanımlamıştır. Aynı şekilde ISO/IEC/ IEEE 15288 Sistem ve Yazılım Mühendisliği – Yazılım Yaşam Döngüsü Süreçleri (Systems and software engineering — System Life Cycle Processes) standardı da benzer bir şekilde yazılım/sistem geliştirme süreçlerini sıralamaktadır.

Yazılım Mühendisliği Bilgi Birikimi Kılavuzu (Guide to the Software Engineering Body of Knowledge) (SWEBOK v3.0) bu teknik süreçleri yazılım gereksinimleri, yazılım tasarımı, yazılım inşası (kodlama), yazılım testi, yazılım bakımı şeklinde beş konu başlığı bazında açıklamıştır. İş Analizi Bilgi Birikimi – İş Analizi Bilgi Birikimi Kılavuzu (Business Analysis Body of Knowledge (BABoK) V3, A Guide to Business Analysis Body of Knowledge) gereksinimleri dört farklı kategoride ele almıştır. Bunlar; İş gereksinimleri: Amaç, hedef gibi neden değişim yapıldığını açıklar ve işletmenin tümünü veya ilgilenen bölümü ilgilendiren üst seviye gereksinimlerdir. Paydaş gereksinimleri: İş gereksinimini karşılamak için gerekli paydaş ihtiyaçlarıdır. İki alt kategoriye ayrılırlar. Fonksiyonel gereksinimler, Fonksiyonel olmayan gereksinimler veya hizmet kalitesi gereksinimleridir. Geçiş gereksinimleri: Geçici olarak geçişi sağlamak için gerekli olan gereksinimlerdir. İş Analizi Bilgi Birikimi – İş Analizi Bilgi Birikimi Kılavuzu (Business Analysis Body of Knowledge (BABoK), A Guide to Business Analysis Body of Knowledge) iş analizini, gereksinimleri ortaya çıkarma, analiz etme, doğrulama, yönetme ve tutarlı bir şekilde iş analizinin temel faaliyetleri olarak kabul etmiştir. Ancak, iş analistlerinin tasarımdan bir şekilde sorumlu olduğunu kabul etmek gerekir. Ne derece derinliğe ineceği içinde bulunduğu projenin durumuna göre değişmektedir. Gereksinimler ihtiyaca odaklanır; tasarımlar çözüme odaklanır. Gereksinimler ve tasarımlar arasındaki ayrım her zaman net değildir. Aynı teknikler her ikisinde de kullanılır. Bir gereksinimden tasarım çıkar, o tasarım bir gereksinimin analiz edilmesini doğurabilir. Dolayısı ile şelale yöntemindeki gereksinim analizi ve yazılım tasarımı kavramlarının beraber ele alınması ve çözümün iteratif bir şekilde her şeyi kapsayacak şekilde ortaya konması gerekmektedir. İş Analizi Bilgi Birikiminin Çevik Uzantısı Kılavuzunda (Agile Extension to the Business Analysis Body of Knowledge- BABOK Guide) çevik yaklaşımların, ürünü küçük parçalara dilimleyerek kademeli olarak değer sağladığı aktarılmaktadır. Bunun yanında çevik yaklaşımları artımlı dağıtım, hızlı geri bildirim, öğrenme ve değişime uyum sağlama olanağı şeklinde aktarmıştır. Ayrıca çevik iş analizi ile sürekli geri bildirimde bulunulur, teslimatı öğrenerek önceliklendirir, israfı minimize eder, müşteri değerini artırır şeklinde katkılarından bahsedilmiştir.

Şelale Yöntemi

Genel olarak yazılım geliştirme süreçlerine baktığımızda aşağıdaki şekil 1'de yer aldığı gibi bir şelale yöntemi ile karşı karşıya kalmaktayız. Öncelikle, gereksinim analizi safhası tamamlanıp ardından buranın çıktısı veya bu aşamada elde edilenler yazılım tasarımı safhasına geçirilir. Yazılım tasarımı safhasında yazılımın kodlama aşamasına gelmeden önceki tasarımı yapılır ve çıktısı kodlama aşamasına gönderilir. Kodlama aşamasında ise projede seçilen yazılım geliştirme diline göre kodlama yapılır. Sonrasında ortaya çıkan ürün test faaliyetlerinin gerçekleştirilmesi için test ortamına alınır. Test ortamında farklı testler yapılır ve bulgular kodlama aşamasına tekrar geri gönderilir. Eğer bu veya bir önceki aşama olan kodlama aşamasında herhangi bir tasarım ile ilgili sorun var ise bulgu yazılım tasarımı aşamasına geri gönderilir ve nerede ne düzeltmesi yapılacak ise düzeltilmiş tasarıma göre tekrar kodlanır ve test aşamasına tekrar gelinir. Test aşamasında tüm testleri geçen yazılım uygulamaya alınmak üzere gerekli altyapı hazırlıkları yapılarak ve kullanıcıya eğitimi verilerek uygulamaya alınır. Ardından yazılımda işletim ve bakım safhası başlar. Bu safhada ise işletim aşamasında yeni fonksiyonlar eklenebilir. Bu durumda bu safhalar en başından itibaren sırası ile yapılır. Bunun yanında işletim aşamasında yazılımda herhangi bir hata olması durumunda da yine en başa dönülür ve hatanın nedeni bulunur ve ona göre gerekli değişiklikler faz faz yapılarak yazılımdaki hata giderilir ve tekrar ilgili yazılım hatasız bir şekilde uygulamaya alınarak işletime verilir.

Şekil 1. Bilişim Sistemi Yazılımı Geliştirme Yaşam Döngüsü – Şelale Yöntemi



Gereksinim analizi kapsamında İş Gereksinimleri Spesifikasyonu (Business Requirements Specification) hazırlanması gerekmektedir. Bu başlıklar işletmeden işletmeye değişmesine rağmen ana başlıklar genel olarak aynıdır. ISO/IEC/ IEEE 29148 Sistem ve Yazılım Mühendisliği –Yaşam Döngüsü Süreçleri – Gereksinim Mühendisliği (Systems and Software Engineering — Life Cycle Processes — Requirements Engineering) standardında bu dokümanın içeriğini ana hatları ile giriş (işin amacı, işin kapsamı, işin genel anlatımı, tanıtımlar, paydaşlar), referanslar, iş yönetim gereksinimleri (iş ortamı, amaç ve hedef, iş modeli, bilgi ortamı) iş operasyonel gereksinimleri (iş süreçleri, iş operasyonel politikaları ve kuralları, iş operasyonel modları, iş operasyonel kalitesi, iş yapısı) kullanıcı gereksinimleri, önerilen sistemin konsepti (operasyonel konsepti, operasyonel senaryo) proje kısıtları, ekler ve kısaltmalardan oluşmaktadır. Paydaş Gereksinimleri Spesifikasyonu ile İş Gereksinimleri Spesifikasyonu içerik olarak benzemektedir.

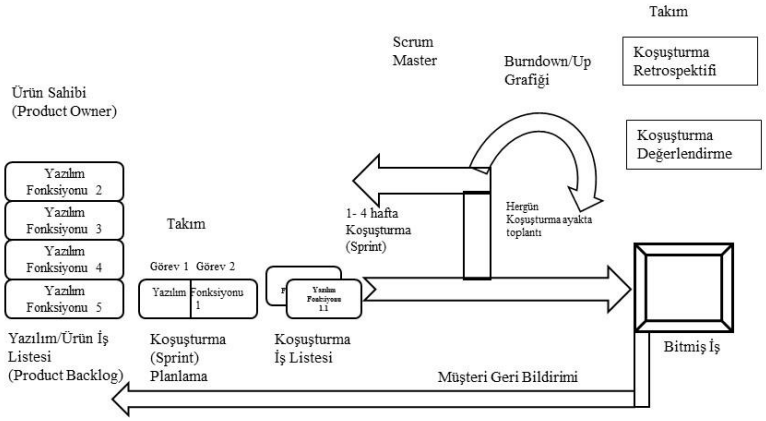
Bunun yanında yazılım tasarımı esnasındaki çalışmaları ISO/IEC/ IEEE 29148 Sistem ve Yazılım Mühendisliği –Yaşam Döngüsü Süreçleri – Gereksinim Mühendisliği (Systems and Software Engineering — Life Cycle Processes — Requirements Engineering) standardı ana hatları ile giriş (amaç, kapsam, ürüne genel bakış, tanımlar) kaynaklar, özel gereksinimler (harici arayüzler, işlevler, kullanılabilirlik gereksinimleri, performans gereksinimleri, mantıksal veri tabanı gereksinimleri, tasarım kısıtlamaları, yazılım sistem özellikleri, destekleyici bilgiler) doğrulama ve ekler şeklinde dokümante edilebileceğini ifade etmiştir.

Scrum Yöntemi

Scrum Kılavuzu- Scrum'ın Tanımlayıcı Kılavuzu Oyunun Kuralları dokümanına göre Scrum, öngörülebilirliği en iyi seviyeye çıkarmak ve riski kontrol etmek için iterasyonlu ve artırımlı (incremental) bir yaklaşım kullanır şeklinde tanımlamıştır. Scrum etkinlikleri dört farklı etkinlik olarak tanımlamıştır. Bunlar, koşuşturma (Sprint) planlama, günlük scrum, sprint değerlendirme, sprint retrospektifidir. Scrum takımı, bir ürün sahibi, geliştirme takımı ve bir de scrum master'dan oluşur. Scrum takımları, kendi kendilerini yönetir ve çapraz fonksiyonludur.

Scrum metodunda iş, şekil 2'de görüleceği üzere öncelikle yazılım/ürün iş listesinin oluşturulması ile başlar. Projelerin durumuna göre bu iş sprint sıfırda yapılır veya bu işlem her bir sprint de normalde ürün sahibi tarafından veya iş analisti rolündeki takım üyesi tarafından yapılır. Scrum takımı farklı rolleri olan ekip üyelerinden oluşur. Yazılım/ürün iş listesi oluşturulduktan sonra artık koşuşturma (sprint) planına başlanabilir. Burada iş, bir veya dört hafta içinde takım üyelerinin iş yapma kapasitelerine göre görevlere bölünür ve sprint planı yapılır. Sprint planı yapıldıktan sonra planda değişiklik yapılması (ilave veya çıkarma) önerilmez. Ancak zorunlu durumlarda iş yapılamıyor duruma geldiğinde değişiklik yapılması mümkündür. Ancak bu değişiklikler takım tarafından yetişmeyecek işleri sprint planından çıkarma şeklinde olmaması gerekmektedir. Acil durumlarda da ilaveler yapılabilmektedir. Sprint ilerleme durumu, ilaveler ve çıkarımların tamamı sprint devam ederken veya sonundaki farklı grafiklerde görülerek izlenebilmektedir. Takım, sprint devam ederken her gün sabahtan, fazla sürmemesi ve farklı konulara girilmemesi için mümkün ise ayakta, günlük toplantılar yapar. Bu toplantıda, dün ne yaptım, bugün ne yapacağım şeklinde tüm takım üyeleri söz alarak konuşur. Sprint sonunda sprint değerlendirme ve retrospektif değerlendirmeleri yapılarak sprint kapatılır. Bu süreçte scrum master rolündeki kişi bu sürecin scrum ilkelerine göre yürütülmesinde takıma yardımcı olan kişidir.

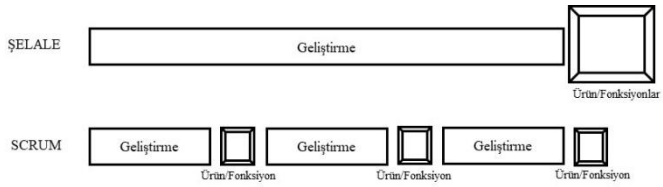
Şekil 2. Bilişim Sistemi Yazılımı Geliştirme Yaşam Döngüsü – Scrum Yöntemi



Şelale ve Scrum Yöntemlerinin Karşılaştırması

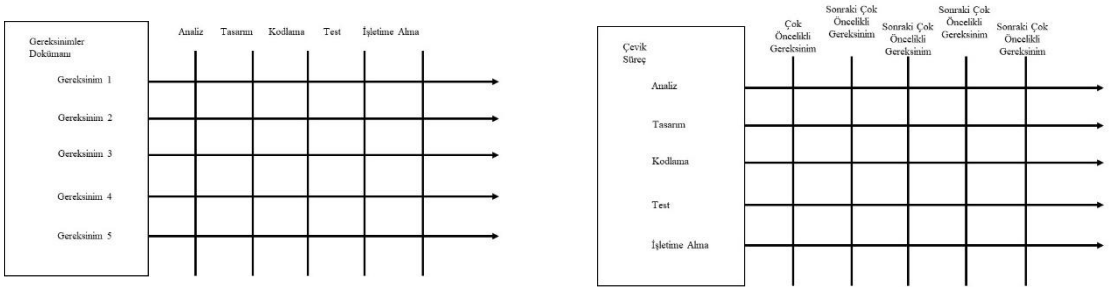
Klasik projeler plan odaklı, çevik yöntemler değişim odaklıdır. Dolayısı ile şelale yöntemindeki kapsam yönetimine baktığımızda kapsam, en başta tanımlanır ve buna göre iş dağılım ağacı çıkarılır. Ardından bu kapsam doğrulanır ve bu kapsamda değişiklik kontrol altında tutulur. Ancak çevik yöntemlerde birikim (backlog) hazırlanır ve bunun üzerinden uygulamaya alma (realese) ve kaç defa tekrarlanacağı planı yapılır. Ardından her bir uygulamaya alma kabul edilir ve sürekli olarak bu adımlarda geri besleme olur. Şekil 3'de her iki yöntemin planlama ve geliştirme durumları gösterilmiştir.

Şekil 3. Şelale ve Scrum Yazılım Geliştirme Planlamaları



Aşağıdaki şekil 4'ün ilk kısmı şelale yönteminin akışını ikinci kısmı ise çevik yöntemlerdeki akışı göstermektedir. Şelale yönteminde gereksinimler sabitlenmiştir ve gereksinimler dokümanı içinde bir bütün olarak her bir safhada onaylanarak akmaktadır. Bunun yanında çevik yöntemde ise durum daha farklı olarak en yüksek öncelikli gereksinimden başlayıp gereksinimler teker teker safhalardan akmaktadır.

Şekil 4. Şelale ve Çevik Yöntemlerde Gereksinimlerin Akışı

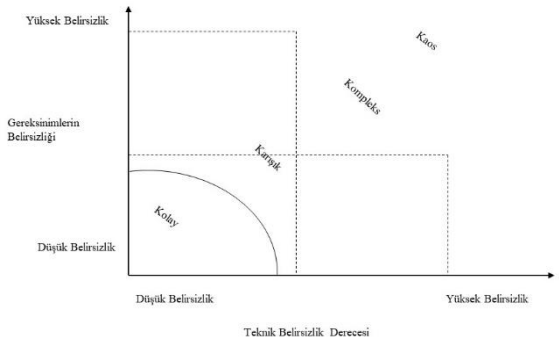


Kaynakça: (Palmquist, Lapham, Miller, Chick ve Ozkaya, 2013)

Projelerde belirsizlik fazla olursa, tekrardan aynı işi yapma riski de artmaktadır. Bu riskin etkisini azaltmak için ekip bu aşamada, bu belirsizliği ele alacak olan küçük ilerlemeli modeli seçer. Ekip, çıkardıkları ürünü doğrulatır ve var ise değişiklikleri yapar ve bir sonrakine geçer. Bu şekilde

gereksinimlerin statik olarak yazılmasına nazaran bu şekilde gidilmesi durumunda müşteri gereksinimleri daha iyi ve hızlı anlaşılır. Belirsizlik artarsa, ekiplerin gereksinimleri statik halde alıp ilerlemeleri durumunda tekrar ele almaları artacak, bu durum zaman kaybına ve maliyete neden olacaktır. Çevik yazılım geliştirmeler (iteratif, artırımlı veya çevik) genellikle araştırma ve geliştirme projelerinde, değişikliğin çok olduğu projelerde, gereksinimlerin belirsiz olduğu veya riskin olduğu projelerde, son hedefin tam olarak tanımlanamadığı projelerde kullanılması uygundur. Şekil 5'de görüleceği üzere hem gereksinimlerde belirsizlik hem de karmaşıklığın yüksek olduğu projelerde kaos yaşanması kaçınılmaz olacaktır.

Şekil 5. Belirsizlik ve Karmaşıklık Modeli



Kaynakça: (Agile Practice Guide PMI, 2017)

Şelale ve scrum yöntemlerindeki iş yapış şekli ile ilgili farklılıklar özet olarak tablo 1'de gösterilmiştir.

Tablo 1. Şelale ve Scrum Yöntemlerinin Karşılaştırılması

Yöntemler	Şelale	Scrum
Konular		
Çalışma şekli	Fonksiyonel gruplar halinde veya takım	Takım
Zamanlama	Seri halinde, birinin çıktısı diğerine gitmektedir	İterasyon halinde, o iterasyonda tüm yazılım geliştirme süreci
Müşteri etkileşimi	Safhalar sonunda onaylama	Döngüler içinde işin içinde bulunma
Değişiklikler	Çok yavaş ve maliyetli yönetilir	Hızlı bir şekilde daha az maliyetle yönetilebilir
Belirginlik	Projedeki belirginlik daha fazla	Duruma göre hareket edildiği için belirginlik az
Planlama	Başında detaylı bir şekilde yapılır	Döngüler halinde yapılır
Gereksinimler	İhtiyaçlar açık ve net ise uygun model	İhtiyaçlar net değil ise uygun model
Teslimat	Teslimatlar daha planlıdır ve bellidir	Teslimatların planlaması zordur ve tam belli değildir
Odak	Projeye odaklanma	Ürüne ve müşteriye odaklanma
Metodoloji	Katı kurallar	Esnek kurallar
Safhalar	Bir kere yaşanır	Yinelemeli şekilde birden fazla yaşanır

Tablodan görüleceği üzere şelale yönteminde proje en başta planlanır. Bu yöntemde ihtiyaçlar açık ve nettir, bir aktivitenin çıktısı diğerine girdi olmaktadır, müşteri ortaya çıkan ürünü veya hizmeti proje sonunda görmekte ve onaylamaktadır. Çevik yazılım geliştirme yöntemi olan scrum da ise yazılım süreci iterayonlu bir şekilde, safhalar yinelemeli bir şekilde birden fazla yaşanmakta ve ürüne ve müşteriye odaklanılmaktadır.

UYGULAMA

İster şelale yönteminde olsun ister çevik yöntemlerden olan scrum'da olsun iş analizi, gereksinimlerin toplanması ve ardından sistem tasarımının çıkarılması aşamalarının bir olarak ele alınarak yazılım modelinin oluşturulması önem kazanmaktadır. Dolayısı ile her iki yöntemde de gereksinimlerin analizi ve sistem tasarımı aşaması vardır. Şelale yönteminde her bir yazılım geliştirme aşamaları yukarıdaki yöntem kısmında açıklandığı gibi bir safhanın bitmesi ve diğer safhanın başlaması şeklindedir. Çevik yöntemlerde ise bu safhalar birden fazla yaşanabilmektedir. İhtiyaç, test edilebilir veya müşteriye teslim edilebilir bir halde geliştirilecek şekilde gereksinim analizi ve yazılım tasarımı aşamasında bölünürse, ardından kodlama ve diğer safhalar ve hatta gereksinim analizi ve yazılım tasarımı aşaması bu bölünme kadar tekrarlanacaktır. Bununla ihtiyacın doğru anlaşılıp anlaşılmadığını ve ona göre üretilen çözümün doğru yolda mı ilerlediğini net bir şekilde gösterecektir. Şelale yönteminde bu safha aslında gereksinim analizi ve yazılım tasarımı safhalılarının sonlarında, müşteriye ilgili gereksinimleri ve tasarımı göstermek sureti ile yaşanmaktadır. Ancak bazı durumlarda geç kalınmış olunabilmektedir. Aslında çevik yöntemlerdeki en büyük sorun işin bölünmesi aşamasında yaşanmaktadır. Çünkü iş neresinden ve nasıl bölünecektir ki müşteriye artırımlı olarak bir ürün sunulabilsin. Her iki yönteminde birbirine karşı olumlu veya olumsuz yanları mevcuttur.

Şelale Yönteminden Scrum Yöntemine Geçiş

Günümüzde artık işletmeler, yazılım dünyasından, ihtiyaçlarını çok hızlı bir şekilde geliştirmelerini ve hatta küçük küçük geliştirmelerin yapılarak uzun süreler beklemeden kullanmak istemektedir. Ayrıca ürünü (sistemi/yazılımı) son aşamada görmek yerine, artırımlı olarak görerek hataların önüne geçmek istemektedir. Ayrıca scrum takımlarının takım içinde uyumu sağlaması durumunda fonksiyonel yapıya göre daha hızlı yazılım geliştirme kabiliyetleri olabilmektedir. Bu ve bunun dışındaki çok farklı nedenlerden dolayı işletmeler scrum takımları kurmak sureti ile şelale yönteminden scrum yöntemine doğru geçiş sergilemektedir.

Şelale yönteminden scrum yöntemine geçiş aşamasında yapılması gereken adımlar aşağıda konu konu ele alınmıştır.

Yazılım Geliştirme Süreci: İlk olarak ele alınması gereken başlıklardan bir tanesidir. Şelale yöntemi işleri ardı ardına yapılmasını gerektirmektedir. Scrum yöntemi ise bunu küçük parçalara bölerek ilerlemeyi hedef almaktadır. Ayrıca şelale yönteminde her bir safha işin durumuna göre çok uzun olacağı için hem olası bir değişiklikte veya müşteriyi yanlış anlamada değişim maliyetli olabilmektedir. Küçük parçalara bölerek her bir iterasyonda müşteriye sunulan yazılım fonksiyonu hem müşterinin önüne hızlı bir şekilde işlerin akması hem de değişiklik olması durumunda onu yönetmek daha da kolay olmaktadır.

Organizasyon Yapısı: Şelale yönteminde organizasyon yapısı genelde fonksiyonel olarak ayrılmıştır. Ana hatlarıyla iş (sistem) analizi grubu, yazılım grubu, test grubu veya farklı proje ihtiyaçlarına göre ilaveler veya bu gruplarda eksiltmeler yapılabilmektedir. Ancak bu şekildeki organizasyon yapısında işlerin ilerlemesi bazı durumlarda çok iyi koordinasyon sağlanmaması durumunda gruplar arası iş aktarımları gecikebilir veya işin gruplar arasında akışı sorun olabilmektedir. Asıl sorun, işin sahibi olmadığından, iş bazen gecikebilmektedir. Scrum yönteminde ise gruplar şeklinde değil tek bir takım ve takım içinde her bir gruptan yeteri kadar personel olması şeklinde yapılanma vardır. Bunun en büyük faydası, takım işi sahiplenmekte ve koordinasyonsuzluk ortadan kalkmaktadır.

Dokümantasyon: Bu alanda da değişim yaşanması gerekmektedir. Çevik yöntemler genelde dokümantasyonun en aza indirmek ve hatta bazıları hiç olmaması yönünde görüşler ortaya koymaktadır. Aslında burada bakış açısı çok önemlidir. Asıl odaklanılması gereken yeteri derinlikte, içerikte ve çeşitlilikte dokümantasyonun olmasıdır. İşe yaramayacak olan dokümantasyonu neden

üretiyoruz felsefesi her zaman çevik yöntemlerde olmuştur. Aslında bu felsefe şelale yönteminde de vardır ancak bazı durumlarda dokümantasyon içinde boğulma durumları yaşanabilmektedir. Çevik yöntemler dokümantasyonu bazı durumlarda kodun içine gerekli açıklamaları yerleştirerek de sağlanabileceğini aktarmaktadır. Bunun faydası da dokümanlarda yaşanan güncelleme sorunlarının bu şekilde çözümlenmek istenmesidir. Güncellenmemiş doküman her zaman işe yaramayan ölü dokümandır. Eğer doküman yaşatılabiliyorsa ve değişiklikler yansıtabiliyorsa o doküman işe yarayacaktır. Çoğu yazılım geliştirme projelerinde görülen en büyük sorunlardan birisi de dokümantasyonu güncel tutmaktır. Bundan dolayı özellikle gereksinimlerin net olmadığı aşamada prototipleme yöntemi dokümantasyon oluşturulmasından ekibi kurtarabilecektir.

Planlama, Yönetim ve Koordinasyon: Scrum yönteminde bu kapsamda dışarıdan çok fazla müdahale olması istenmemektedir. İşler, iş listesinde (backlog) bekler ve takımın buradan işi belirlenmiş olan önceliğe göre alması istenmektedir. Ancak bazı durumlarda takımlara yol göstermek ve işleri önceliklendirmek ve takım içindeki dinamikleri iyi şekilde oturtmak gerekmektedir. Bu açıdan dışarıdan bir destek gerekebilmektedir.

İş Gereksinimlerinin Toplanması ve Yazılım Tasarımı: Yazılım geliştirme yaşam döngüsünün adımlarından olan iş gereksinimlerinin toplanması ve yazılım tasarımı aşaması hem şelale hem de scrum yönteminde de büyük bir önem arz etmektedir. Her iki yöntemde bu aşama şekilsel olarak farklıdır ancak temelinde bu aşamada yapılmak istenen müşterinin ihtiyacı olan gereksinimleri belirleyip onları yazılım tasarımına döndürebilmektir. Sonuç olarak bu aşamada iş yapış şekillerini prototip tabanlı bir yapıya döndürmek en büyük faydayı sağlayacaktır. İş analistinin en büyük görevlerinden birisi müşteri ihtiyaçlarının doğru bir şekilde alınması ve bunların sistem tasarımı haline döndürülmesidir. İş analisti, iş dünyası ile yazılım geliştirme dünyası arasında köprü görevi yapmaktadır. Bu köprü görevini yerine getirirken hem iş dünyasından gereksinimleri alabilmek için hem de bunu yazılım dünyasındaki yazılımcılara aktarabilmek için bu aşamada çok farklı yöntemler kullanmaktadır. Bunlardan en önemlisi prototiplemedir. Bu araç hem şelale yönteminin hem de scrum yönteminin en büyük tekniklerinden birisidir.

Şelale yönteminde gereksinim analizi ve yazılım tasarımı aşaması ile scrum yönteminde yazılım iş listesi (Product backlog) aşamasının nasıl oluşturulduğunu karşılıklı olarak ele alıp bu aşamadaki bulgulara aşağıda yer verilmiştir.

Şelale yönteminde gereksinimler, müşteriler, istek sahibi veya ihtiyaç sahibi ile görüşülerek ortaya çıkarılmaktadır. Bu aşamada gereksinimleri almak için gözlem, toplantı, görüşme, doküman inceleme, süreç akışı bazlı bilgi edinme, senaryo eşliğinde iş yapışı inceleme/dinleme şeklinde farklı teknikler kullanılabilmektedir. Bu çalışmaların ardından gereksinimler listelenmekte, önceliklendirilmekte, fonksiyonel ve fonksiyonel olmayan gereksinimler şeklinde bölümlendirilmektedir. Bu çalışma bize tamamen gereksinimleri sağlamış olmaktadır. Bunun ardından yazılımın gerçek anlamda yukarıda açıklandığı üzere hem müşteri hem de takımdaki yazılımcının anlayacağı şekilde dokümante edilmesi veya aktarılması gerekmektedir. Bu aşamada artık yazılım tasarımı başlamaktadır. Şelale yönteminin ikinci aşaması yazılım tasarımı aşamasının amacı elde etmiş olduğumuz gereksinimleri karşılayacak yazılımın tasarımını yapmak olacaktır. Yazılım tasarımının ilk aşamasında yazılımın genel olarak tasarımının fonksiyon fonksiyon anlatılmasıdır. Yazılımın her bir fonksiyonunda kısıtlar, varsayımlar, hata durumunda verilecek mesajlar ve yazılımın nasıl davranacağı gibi tüm hususlar göz önüne alınarak aktarılmalıdır. Bu aşamada farklı yöntemler kullanılabilmektedir.

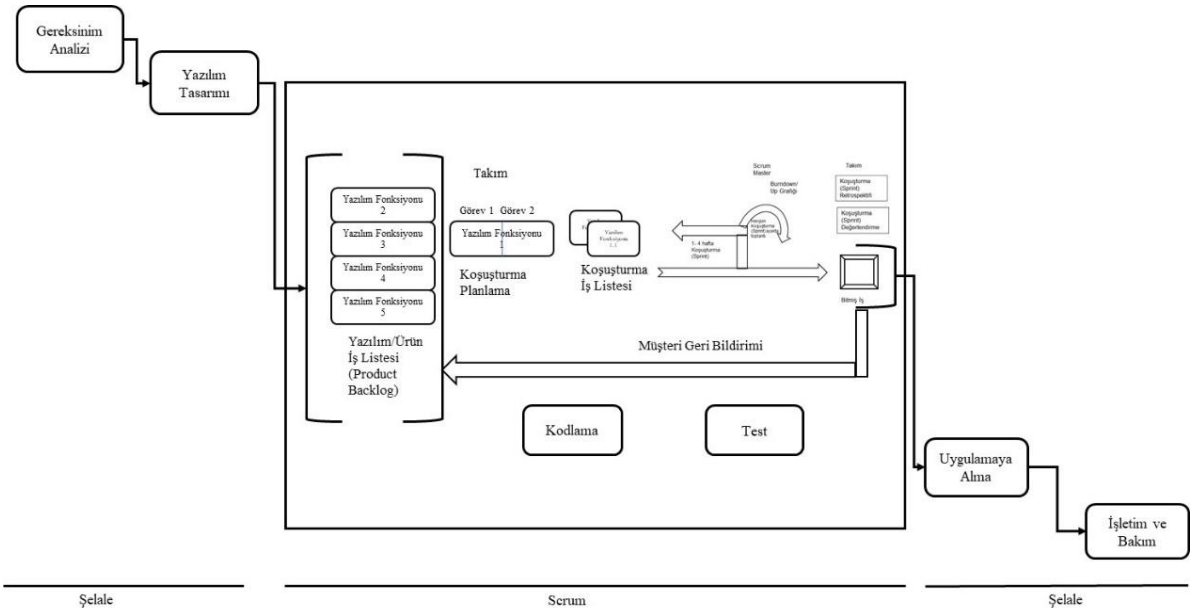
Scrum yönteminde yazılım/ürün iş listesini oluşturmak da aslında bir çeşit kullanıcı ihtiyaçlarından yazılımın gereksinimlerini çıkarmak ve bunların bir şekilde tasarımını yapmaktır. Bu aşamada en çok kullanılan tekniklerden bir tanesi kullanıcı senaryosudur (user story). Çevik yöntemlerde şelale yöntemi gibi çok fazla dokümantasyona gidilmesi pek istenmemektedir. İhtiyacı görecek kadar çeşitlilik ve derinlikte dokümantasyon mantığı ile ilerlenilmektedir.

Karma Yöntem

Projelerin içeriği, projede çalışanların bilgi ve birikimleri, yönetimin yaklaşımı, gereksinimlerin belirgin olup olmaması, proje büyüklüğü gibi birçok farklı neden projelerde kullanılan yöntemi etkilemektedir. Projelerin durumlarına göre şelale, bazı durumlarda ise scrum yöntemini kullanmak projelerin başarısını etkileyecektir. Projelerin yöntemine karar vermeden önce projelerin

içerikleri net olarak incelenmeli ve ona göre en fazla katma değeri sağlayacak olan yöntem seçilmelidir. Bazı durumlarda da sadece bir yöntem yerine hem şelale yönteminin hem de scrum yönteminin en iyi yanlarını alarak karma melez bir model uygulamak projelerin başarısı için faydalı olabilecektir. Bu durumda da yine projenin özellikleri irdelenmeli ve ona göre karma yöntemin kullanılması değerlendirilmelidir. İşletmelere baktığımızda bu şekilde uygulamaların var olduğunu görebilmekteyiz. Aşağıda, şekil 6'da karma yöntemin nasıl olabileceği gösterilmektedir.

Şekil 6. Şelale ve Scrum Yöntemlerinin Yazılım Geliştirme Yaşam Döngüsünde Karma Kullanımı



Gereksinim analizi ve yazılım tasarımı aşaması şekil 6'dan da görüleceği üzere şelale yöntemi ile ilerlenir ve bu şekilde gereksinimlerin tamamı elde edilerek sistem tasarımı da tamamlanıp her bir sistem tasarımı yazılım fonksiyonlarına bölünüp scrum yönteminde kullanılabilecek halde küçültülür. Kodlama ve test aşamaları scrum yöntemi ile yapılır ve en sonunda ise uygulamaya alma ile işletim ve bakım safhalarında da şelale yöntemi uygulanır.

Bu karma yöntem özellikle gereksinimlerin ve sistem tasarımının net olmadığı ve bunu belirlemek için zaman kaybedilmesi durumu var ise ve ardından her bir sprinte belirlenmiş olan yazılım fonksiyonlarının alınıp onun kodlamasının yapılmasının uygun olacağı yapılarda daha iyi çalışacaktır. Özellikle sprint içinde gereksinim belirlemek ve ardından bu gereksinimleri sprint içinde kodlamak ve test etmek mümkün olmadığı durumlarda daha iyi çalışacaktır. Çünkü bu karma yöntemde özellikle sistem analizi ve tasarımı safhası kendi başına çalışmakta ve bunun sonucunda çıkan iş listeleri gruplanmakta ve kullanıcıya dağıtımı yapılabilecek olan yapılarda sprint'lere bölünmekte ve hatta farklı scrum takımları tarafından da geliştirme çalışmaları yapılabilmektedir. Eğer sprint içinde sistem analiz ve tasarımı yapılması durumunda bu sürenin belirsiz olabileceği nedeni ile sprint içinde planlanmış olan işlerin bitirilememe durumu söz konusu olabilmektedir. Bazı durumlarda eğer sprint içinde yapılacak olan sistem analizi ve tasarımı safhasındaki işler net olarak tamamlanabilecek durumda ise bu yöntemin bir miktar daha genişletilmesi ve yazılım tasarımı safhasının da kapsamın içine alınması söz konusu olabilmektedir. Özellikle çok büyük projelerde sistem analizi ve tasarımı aktivitelerinin önceden yapılarak işin bölümlenmesi ve ona göre çalışmaların başlaması uygun olacaktır.

Karma yöntem özellikle sistem tasarımının yaptırılıp kodlama işlemlerinin farklı takımlara bölünmesi durumunda ele alınabileceği durumda olan işletmeler için uygun olabilecektir. Fonksiyonlar işletmenin kendisi tarafından belirlenebilir ve kodlama işlemeleri istenildiği kadar scrum takımlarına bölünerek kodlama işlemleri gerçekleştirilebilir. Hatta sprint içinde çalışacak olan takımların aynı yerde olmasına da gerek olmayabilir.

Önerilen karma yöntemin uygulandığı projeler olarak bakıldığında ise; bu yöntem özellikle şelale yönteminden scrum yöntemine geçiş yaparken işletmelerin yaşayabileceği geçiş evresinde ele alınan farklı proje çalışmalarında görülmüş ve ona göre ortaya konulmuştur. Özellikle işletmeler birdenbire scrum yöntemine tüm unsurlarını geçirmede zorluklar çekmektedir. Sistem analizi ve tasarımının net olarak çizilmesi beklenmekte ve ardından kodlama ve test aktiviteleri şelale yönteminde olduğu gibi beklenmektedir. İşte bu gibi durumlarda bu yöntem sayesinde hem şelale hem de scrum yönteminin karması olan yöntem ortaya çıkabilmektedir. Hatta yöntemin işletmede kullanımı kalıcı hale gelebilmektedir. Şöyle ki, ürün iş listeleri sürekli olarak kullanıcılar ile beraber belirlenmekte ve ardından da kodlama ve test aktiviteleri içine scrum yöntemi ile dahil edilerek geliştirme kısmı hızlanabilmektedir. Bu anlamda hem şelale yönteminden scrum yöntemine geçiş yapan işletmelerde hem de sistem analizi ve tasarımı safhasının belirsiz olduğu projelerde bu yöntem ile karşılaşmak mümkün olabilmektedir.

SONUÇ

Yazılım geliştirme yaşam döngüsünde tek bir standart model olması mümkün değildir. İster şelale yöntemi kullanılsın ister scrum yöntemi kullanılsın bir şekilde yöntemin bir yerinde bir değişiklik yapılmış ve uygulayan kendine göre uyarlamıştır. Projelerin içeriğine göre uygun olan bir yöntemin seçilmesi tabiki en uygun çalışma şekli olacaktır. Ancak, günümüzde artık şelale yöntemini bırakıp çevik yöntemlere doğru bir geçiş süreci çok hızlı bir şekilde yaşanmaktadır. Dolayısı ile bu ihtiyaç, şelale yönteminin sorunlarından kaçmak veya bazen de çevik yöntemlerin avantajlarından faydalanmak için kaynaklanmaktadır. Bazende tek bir yöntem kullanmak yerine birden fazla yöntemin birleştirilerek karma bir yöntem elde edilebilmekte ve tek bir yönteme göre daha fazla fayda elde edilebilmektedir. Ancak amaç, hangi yöntem kullanılır ise kullanılsın müşteriye zamanında, kaliteli ve istediği sistemi teslim etmek olması gerekmektedir.

Bu çalışma ile özellikle şelale yönteminden scrum yöntemine geçişte ele alınması gereken hususlar vurgulanarak uygulayıcıların dönüşümü yapabilmeleri için bir rehber niteliğinde bilgi sunulmak istenmiştir. Özellikle değişime açık olmayan veya nerede ne değişim yapılması gerektiğini bilmeyen veya değişimin sonucunda ne ile karşılaşacağını bilmeyen uygulayıcılar veya araştırmacılar bu çalışmada yer alan geçiş aşamalarındaki bilgiler ile daha kolay ve korkusuzca daha çevik yöntemlere geçiş sağlayabileceklerdir. Ayrıca, her firma veya proje kapsamında yer alan yazılım geliştirme yöntemi kendine has bazı hususları içinde barındırmaktadır. Ne tam anlamı ile şelale ne tam anlamı ile scrum nede tam anlamı ile tek başına başka bir yöntemi uygulamaktadırlar. Genelde çoğu firma veya projede firmanın veya projenin dinamiğine uygun bir şekilde karma yöntem uygulanmaktadır. Bu çalışma kapsamında da bu vurgulanmış ve şelale ve scrum yöntemlerinin en iyi adımlarını alarak kendine has bir karma yöntem önererek geliştirici ve araştırmacılara bilgiler sunmak hedeflenmiştir. Literatürdeki çalışmalar incelendiğinde şelale yönteminden scruma geçiş veya her ikisinin karmasını açıklamaya çalışan bir çalışmaya rastlanmamıştır. Genelde yöntemler karşılaştırılmıştır. Makalenin bu boşluğu doldurması hedeflenmiştir.

İlerleyen zamanda bu çalışmaya ilave olarak scrum yöntemi ile daha farklı yöntemlerin karşılaştırması yapılabilir. Ayrıca scrum yönteminden farklı veya uzantısı şeklinde ileride ortaya çıkabilecek farklı yöntemlerin araştırması da yapılabilir.

KAYNAKÇA

Agile Extension to the BABOK Guide (2017). International Institute of Business Analysis (IIBA) Agile Alliance.

Agile Practice Guide (2017). Project Management Institute, Inc.

AgileTurkey (2019). 8th Annual Agility Report. Türkiye, Rap. Agile Turkey:1.

Andrei B., Casu-pop B., Gheorghe S., Boiangiu C. (2019). A Study On Using Waterfall And Agile Methods In Software Project Management, Journal Of Information Systems & Operations Management, 13 (1) 125-135.

Aydıner S. A., Esen F. M., Özlü E. (2020). Türkiye'de Çevik Yazılım Geliştirme Süreçlerinde Scrum Yöntemini Uygulayan İşletmelerin Başarı Faktörleri. Bilişim Teknolojileri Dergisi, 13 (4): 463-477.

Balaji S. ve Murugaiya M. S. (2012). Waterfall vs. V-Model vs Agile: A comparative study on SDLC. International Journal of Information Technology and Business Management, 2(1): 26-30.

Başar A., Özkaya A., Kesgin F. (2015) Yazılım Geliştirme Süreçlerinde Şelale Yönteminden Çevik Yaklaşıma Geçiş: Bir Teknoloji Şirketinde Uygulama, IX. Ulusal Yazılım Mühendisliği Sempozyumu, İzmir, 224-231.

Beck K., Beedle M. ve Bennekum A. (2001). Çevik Yazılım Geliştirme Manifestosu. http://agilemanifesto.org/iso/tr/manifesto.html (01 Mayıs 2020).

Beck K., Beedle M., Bennekum A., Cockburn A., Cunnigham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R. C., Mellor S., Schwaber K., Sutherland J., Thomas D. (2001). Çevik yazılımın oniki prensibi. http://agilemanifesto.org/iso/tr/principles (01 Mayıs 2020).

Bhavsar K., Shah V. ve Gopalan S. (2020). Scrumbanfall: An Agile Integration of Scrum and Kanban with Waterfall in Software Engineering. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 9(4): 2075-2084.

Boehm B. ve Turner R. (2003). Rebalancing Your Organization's Agility and Discipline, Third XP and Second Agile Universe Conference, New Orleans, USA.

Business Analysis Body of Knowledge (BABoK) A Guide to Business Analysis Body of Knowledge (2015). International Institute of Business Analysis.

Ceylan Z. ve Gürsev S. (2020) AHP ve TOPSIS Yöntemleri ile Bilgi Teknolojileri Projelerinde Scrum-Kanban-Şelale Uygulamaları Karşılaştırması. Bilişim Teknolojileri Dergisi, 13 (3): 329-339.

Davis B. ve Radford D. (2014). Going Beyond The Waterfall Managing Scope Effectively Across the Project Life Cycle içinde 145-160. Plantation, Florida, USA, J.Ross Publishing.

Forrester (2011). Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today- Manage The Water-Scrum And Scrum-Fall Boundaries To Increase Agility Report., Cambridge, USA, Rep. Forrester:262011.

Gencer C. ve Kayacan A. (2017). Yazılım Proje Yönetimi: Şelale Modeli ve Çevik Yöntemlerin Karşılaştırılması. Bilişim Teknolojileri Dergisi, 10 (3): 335-352.

Guide to the Software Engineering Body of Knowledge (SWEBOK) (2014). IEEE Computer Society.

Hewlett Packard (HP) Enterprise. (2017). Agile is the New normal, Adopting Agile Project management. (01 Mayıs 2020) https://softwaretestinggenius.com/docs/4aa5-7619.pdf.

Kuhrmann M., Diebold P., MüNch J., Tell P., Garousi V., Felderer M., Trektere K., McCaffery F., Linssen O., Hanser E., R. Prause C. (2017). Hybrid software and system development in practice: waterfall, scrum, and beyond, ICSSP 2017: Proceedings of the 2017 International Conference on Software and System Process, 30–39

Macit Y. Ve Tüzün E. (2015). Uygulama Yaşam Döngüsü Yönetimi Karşılaştırmalı Süreç İncelemesi. Turkish National Software Engineering Symposium. UYMS 2015. İzmir. 122-133.

Mahalakshmi M. ve Sundararajan D. M. (2013). Traditional SDLC vs Scrum Methodology – A Comparative Study. International Journal of Emerging Technology and Advanced Engineering, 3(6): 192-196.

Palmquist M. S., Lapham M. A., Miller S., Chick T. ve Ozkaya I. (2013) Parallel Worlds: Agile And Waterfall Differences And Similarities, 1. baskı, Massachusetts, ABD: Carnegle Mellon University.

Royce W. W. (1970). Managing The Development of Large Software Systems, IEEE WESCON, 1 (2): 1-9.

Schwaber K. ve Sutherland J. (2017) Scrum Kılavuzu- Scrumın Tanımlayıcı Kılavuzu: Oyunun Kuralları, Türkçe'ye çeviren Agile Turkey.

Singhto W. ve Phakdee N. (2016). Adopting a combination of Scrum and Waterfall methodologies in developing Tailor-made SaaS products for Thai Service and manufacturing SMEs. 2016 International Computer Science and Engineering Conference (ICSEC), Chiang Mai, Thailand.

Software Extension To The PMBOK Guide (2013). Project Management Institute, IEEE Computer Society.

Sommerville I. (2016). Software Engineering, Harlow, England, Pearson.

Stoica M., Ghilic-Micu B., Mircea M., Uscatu C. (2016). Analyzing Agile Development – from Waterfall Style to Scrumban, Informatica Economica, 20(4), 5-14.

Systems and software engineering — Life cycle processes (2018). Requirements engineering ISO/IEC/ IEEE 29148.

Systems and software engineering — Life cycle processes (2011). Requirements engineering ISO/IEC/ IEEE 29148.

Systems and software engineering — Software life cycle processes (2017). ISO/IEC/ IEEE 12207.

Systems and software engineering — System life cycle processes (2015). ISO/IEC/ IEEE 15288.

The Standish Group (2018). Chaos Report. The Standish Group, USA, Rep. The Standish Group: 1.

Vohra P., Singh A. (2013). A contrast and comparison of modern software process models. International Conference on Advances in Management and Technology, Patiala, Hindistan.